WRDC-TR-90-8007
Volume VII
Part 4

INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume VII - Communications Subsystem
Part 4 - IBM IHC and IPC Development Specification

S. Barker

Control Data Corporation
Integration Technology Services
2970 Presidential Drive
Fairborn, OH 45324-6209

September 1990

Final Report for Period 1 April 1987 - 31 December 1990

Approved for Public Release; Distribution is Unlimited

MANUFACTURING TECHNOLOGY DIRECTORATE
WRIGHT RESEARCH AND DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6533

[illegible]

DAVID L. JUDSON, Project Manager
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

FOR THE COMMANDER:

[illegible]

BRUCE A. RASMUSSEN, Chief
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

Unclassified

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release; Distribution is Unlimited.
4. PERFORMING ORGANIZATION REPORT NUMBER(S) DS 620343300	5. MONITORING ORGANIZATION REPORT NUMBER(S) WRDC-TR-90-8007 Vol. VII, Part 4
6a. NAME OF PERFORMING ORGANIZATION Control Data Corporation; Integration Technology Services 6b. OFFICE SYMBOL *(if applicable)*	7a. NAME OF MONITORING ORGANIZATION WRDC/MTI
6c. ADDRESS *(City, State, and ZIP Code)* 2970 Presidential Drive Fairborn, OH 45324-6209	7b. ADDRESS *(City, State, and ZIP Code)* WPAFB, OH 45433-6533
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Wright Research and Development Center, Air Force Systems Command, USAF 8b. OFFICE SYMBOL *(if applicable)* WRDC/MTI	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUM. F33600-87-C-0464
8c. ADDRESS *(City, State, and ZIP Code)* Wright-Patterson AFB, Ohio 45433-6533	10. SOURCE OF FUNDING NOS.
11. TI[illegible] IBI[illegible] See block 19	

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
78011F	595600	F95600	20950607

12. PERSONAL AUTHOR(S)
Structural Dynamics Research Corporation: Barker, S., et al.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Yr.,Mo.,Day)*	15. PAGE COUNT
Final Report	4/1/87-12/31/90	1990 September 30	83

16. SUPPLEMENTARY NOTATION
WRDC/MTI Project Priority 6203

17. COSATI CODES

FIELD	GROUP	SUB GR.
1308	0905	

18. SUBJECT TERMS *(Continue on reverse if necessary and identify block no)*

19. ABSTRACT *(Continue on reverse if necessary and identify block number)*

This development specification explains the basic architecture, the module linked to the Communications subsystem (COMM), the Network Transaction Manager (NTM) and APS, Inter-Process, Primitives, Process Control Primitives, PRC Support Routines, Environment Control Modules, Inter-Host Primitives, Interface to CICS Application, and CICS Interface Primitives.

BLOCK 11:

INTEGRATED INFORMATION SUPPORT SYSTEM
Vol VII - Communications Subsystem

Part 4 -IBM IHC and IPC Development Specification

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED x SAME AS RPT. DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL David L Judson	22b. TELEPHONE NO. *(Include Area Code)* (513) 255-7371 — 22c OFFICE SYMBOL WRDC/MTI

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

DS 620343300
30 September 1990

FOREWORD

This technical report covers work performed under Air Force Contract F33600-87-C-0464, DAPro Project. This contract is sponsored by the Manufacturing Technology Directorate, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio. It was administered under the technical direction of Mr. Bruce A. Rasmussen, Branch Chief, Integration Technology Division, Manufacturing Technology Directorate, through Mr. David L. Judson, Project Manager. The Prime Contractor was Integration Technology Services, Software Programs Division, of the Control Data Corporation, Dayton, Ohio, under the direction of Mr. W. A. Osborne. The DAPro Project Manager for Control Data Corporation was Mr. Jimmy P. Maxwell.

The DAPro project was created to continue the development, test, and demonstration of the Integrated Information Support System (IISS). The IISS technology work comprises enhancements to IISS software and the establishment and operation of IISS test bed hardware and communications for developers and users.

The following list names the Control Data Corporation subcontractors and their contributing activities:

SUBCONTRACTOR	ROLE
Control Data Corporation	Responsible for the overall Common Data Model design development and implementation, IISS integration and test, and technology transfer of IISS.
D. Appleton Company	Responsible for providing software information services for the Common Data Model and IDEF1X integration methodology.
ONTEK	Responsible for defining and testing a representative integrated system base in Artificial Intelligence techniques to establish fitness for use.
Simpact Corporation	Responsible for Communication development.

Structural Dynamics Research Corporation	Responsible for User Interfaces, Virtual Terminal Interface,and Network Transaction Manager design, development, implementation, and support.
Arizona State University	Responsible for test bed operations and support.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

SECTION 1

BASIC ARCHITECTURE OF THE IBM INTERFACE

The IBM Interface consists of the following programs:

1. The OSIIBM stub, which must be link-edited with all programs in the system, including the COMM, NTM, QP, and AP programs. It contains entry points for all the primitive routines and invokes the proper interface program to perform the requested function. It also contains the primitive routine ENDRUN.

2. The OSIIPC program which contains:

   o CRTMBX, for creating a mailbox

   o DELMBX, for deleting a mailbox

   o SNDMSG, for sending a message to a mailbox

   o RCVMSG and GETMSG, for receiving a message from a mailbox

   o SETTIM, for setting a timer

   o CNLTIM, for cancelling a timer, and

   o WAIT01 thru WAIT22, for waiting for completion of one event out of a list of events

3. The OSIPRC program which contains:

   o CRTPRC, for creating a process (subtask)

   o DELPRC, for deleting a process

   o GETNAM, for providing a process with its own process name

4. The OSIIHC program (and its error exit IHCEXT) which contains:

   o INILAN, for initiating communication with the VAX or Level 6

- TRMLAN, for terminating communication with the with the VAX or Level 6
- XMTLAN, for sending a message to the VAX or Level 6
- RCVLAN and GETLAN, for receiving a message from the VAX or Level 6
- CNLLAN, for cancelling a receive from the VAX or Level 6

5. The VTAM control block modules, which are assembled from macro calls, and which contain the VTAM control blocks and other data required to communicate with the VAX or Level 6. Two of these modules must be created -- one for the VAX and one for the Level 6. The load module names are the same as the port names.

6. The error logging programs:

   - OSIERR, which contains ERRPRO, which formats and time-stamps error messages and sends them to the error log mailbox
   - ERRLOG*, which writes error messages to the error log (* this program will be implemented by GE)

7. The PRC support routines:

   - OSIATCH, the ATTACH stub, which is the first program to get control when a subtask is created; it obtains the work areas required by the IBM Interface programs
   - OSIETXR, the End-of-Task exit, which searches for and removes any remaining table entries for a subtask when it ends

8. The environment control modules and tables:

   - OSIMVSI, which is the first program executed in the address space, and which initializes the environment
   - OSICOMTB, the COMMON TABLE, which contains other tables with information required by all the primitives, as well as data required to initialize the address space. The COMMON TABLE is assembled from macro calls.

All the IBM interface modules are coded in Assembler, and all are re-entrant. The COMMON TABLE and the VTAM control block modules must also be link-edited as re-entrant, even though they are in fact modified during execution.

NOTE: The following modules are part of the IBM Interface, but they are not being implemented by On-Line Software. For this reason, they are not described in this document.

9. The Console Primitive program which handles:

   o Initiating and terminating communication with a terminal

   o Sending and receiving messages to/from a terminal

** As a temporary measure, an interface program is being provided which accepts the console primitive calls and invokes the IHC program to communicate with a terminal.

10. The modules required to run a transaction in a CICS region

   o The Psuedo-AP

   o The CICS Interface program

   o The 3270 Emulator program

   o The VTAM SLU control block table with entries for as many terminals as are concurrently being emulated

SECTION 2

MODULE LINKED TO COMM, NTM AND APS

2.1 OSIIBM

Entry from the IISS TEST BED programs into the IBM interface is accomplished through COBOL calls which are resolved in a stub module (OSIIBM) which must be link-edited to each COMM, NTM and AP program. This module has multiple entry points, one for each primitive which a program can call.

The entry points are:

CRTMBX	SETTIM	ERRPRO	CRTPRC	COMM only:
SNDMSG	CNLTIM	ENDRUN	DELPRC	INILAN
RCVMSG	WAIT01		REQPRC	XMTLAN
GETMSG	thru			RCVLAN
DELMBX	WAIT22			GETLAN
RELEVB				CNLLAN
				TRMLAN

FUNCTIONS PERFORMED:

A. RECEIVE CONTROL AT AN ENTRY POINT

1. Save the COBOL program's registers

2. Record the type of call

B. SET UP TO PASS CONTROL

1. Locate the Task Work Area (TWA) by searching back through the save area chain

2. Save in the TWA:
   type of call
   address of the parameter list (from the COBOL program)

C. LINK TO THE PRIMITIVES

1. If an IPC, PRC, or IHC call,
   LINK to appropriate interface program

```
    2.  If an ERRPRO call,
          set up ERRPRO parameter list in the TWA LINK
          to OSIERR

    3.  When control is returned,
          restore the COBOL program's registers
          RETURN to caller

D.  TERMINATE THE COBOL PROGRAM

    1.  If an ENDRUN call,
          restore the ATTACH stub's registers (from the
          TWA)
          RETURN to ATTACH stub
             (process is ended)
```

SECTION 3

INTER-PROCESS PRIMITIVES (IPC)

3.1 Create Mailbox

The CRTMBX primitive routine is in program OSIIPC.

```
CALL 'CRTMBX' USING INPUT-MAILBOX-NAME
                    MAILBOX-SIZE
                    EVENT-BLOCK-nn
                    STATUS.
```

FUNCTIONS PERFORMED:

```
A.  VALIDATE PARAMETERS

    1.  If mailbox-name has embedded blanks
        set 'invalid mailbox name' STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

    2.  If mailbox-size is zero
        set 'mailbox size zero' STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

    3.  If mailbox-size is not numeric or is greater than
            the max
        set 'mailbox size greater than maximum'
            STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

    4.  If EVENT-BLOCK is not all zeros
        set 'event block not initialized' STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

B.  CHECK IF MAILBOX ALREADY EXISTS

    1.  Serialize use of mailbox and MAILBOX TABLE

    2.  Load address of COMMON TABLE from TWA

    3.  Search for mailbox-name in MAILBOX TABLE
```

```
          search til high-water-mark
          some slots may be empty
          follow chain-address if any
          if found empty slot
              save address of first empty slot

      4.  If found mailbox-name
          set 'mailbox already exists' STATUS-CODE
          LINK to ERRPRO to log error
          release use of mailbox
              and MAILBOX TABLE
          RETURN to caller

  C.  ESTABLISH MAILBOX

      1.  Allocate MAILBOX TABLE entry
              if no empty slot found
              GETMAIN storage for continuation of MAILBOX
                 TABLE
              chain and format storage obtained
              LINK to ERRPRO to record overflow

      2.  Create mailbox and MAILBOX TABLE entry
          GETMAIN storage of mailbox-size
          format mailbox header
          put mailbox-name, A(storage), A(ECB) in MAILBOX
              TABLE entry
          save address of MAILBOX TABLE entry in EVENT-BLOCK

  D.  PRIME EVENT-BLOCK AND OTHER REQUIRED AREAS

      1.  Move the mailbox-name to the EVENT-BLOCK
          (marks it as the EVENT-BLOCK for this task's input
          mailbox)

      2.  Set EVENT-TYPE to 01 (RECEIVE)

  E.  RETURN CONTROL TO CALLER

      1.  Set 'successful completion' STATUS-CODE

      2.  Release use of mailbox and MAILBOX TABLE

      3.  RETURN to caller
```

3.2 Send Message

The SNDMSG primitive routine is in program OSIIPC.

```
CALL 'SNDMSG' USING TARGET-MAILBOX-NAME
                    BUFFER
                    NUMBER-OF-BYTES
                    EVENT-BLOCK-nn
                    STATUS.
```

FUNCTIONS PERFORMED:

```
      A.   VALIDATE PARAMETERS

           1.   If mailbox-name has embedded blanks
                set 'invalid mailbox name' STATUS-CODE
                LINK to ERRPRO to log error
                RETURN to caller

           2.   If number-of-bytes is zero
                set 'number of bytes zero' STATUS-CODE
                LINK to ERRPRO to log error
                RETURN to caller

           3.   If number-of-bytes is not numeric or is greater
                    than the max
                set 'number of bytes greater than maximum'
                    STATUS-CODE
                LINK to ERRPRO to log error
                RETURN to caller

      B.   CHECK EVENT-BLOCK

           1.   If EVENT-BLOCK is not all zeros
                set 'event block not initialized' STATUS-CODE
                LINK to ERRPRO to log error
                RETURN to caller

      C.   CHECK IF TARGET MAILBOX EXISTS

           1.   Serialize use of mailbox

           2.   Load address of COMMON TABLE from TWA (contains
                MAILBOX TABLE)

           3.   Search for mailbox-name in MAILBOX TABLE
                search til high-water-mark
                follow chain-address if any
                may be empty slots

           4.   If not found mailbox-name
                set 'mailbox not found' STATUS-CODE
                LINK to ERRPRO to log error
                release use of mailbox
                RETURN to caller
```

```
          5.  Mailbox found
              save MAILBOX TABLE entry address in EVENT-BLOCK
                  (to save future search)

      D.  PUT MESSAGE IN TARGET MAILBOX

          1.  Pick up address of storage for mailbox

          2.  If message doesn't fit in mailbox
              set 'mailbox full' STATUS-CODE
              LINK to ERRPRO to log error
              release use of mailbox
              RETURN to caller

          3.  Put message in mailbox

          4.  Adjust current byte-count and displacements

          5.  POST the ECB in the target mailbox EVENT-BLOCK
                  (ECB address is in MAILBOX TABLE entry)

      E.  RETURN CONTROL TO CALLER

          1.  Set "successful completion' STATUS-CODE

          2.  Release use of mailbox

          3.  RETURN to caller
```

3.3 Receive Message

The RCVMSG primitive routine is in program OSIIPC.

```
CALL 'RCVMSG' USING  INPUT-MAILBOX-NAME
                     EVENT-NUMBER
                     EVENT-BLOCK-nn
                     STATUS.
```

FUNCTIONS PERFORMED:

```
      A.  VALIDATE PARAMETERS

          1.  If event-number is zero
              set 'event number zero' STATUS-CODE
              LINK to ERRPRO to log error
              RETURN to caller

          2.  If event-number is not numeric or is greater than
                  maximum
              set 'event number greater than maximum'
                  STATUS-CODE
              LINK to ERRPRO to log error
              RETURN to caller
```

```
B.  CHECK EVENT-BLOCK

    1.  If mailbox-name doesn't match that in EVENT-BLOCK
        set 'invalid event block for mailbox named'
            STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

    2.  If EVENT-TYPE is not 01 (RECEIVE)
        set 'not a receive event block' STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

    3.  If EVENT-OUTSTANDING is 01 (event is outstanding)
        set 'only one outstanding receive permitted'
            STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

C.  CHECK IF MAILBOX EXISTS

    1.  Serialize use of mailbox

    2.  Load address of COMMON TABLE from TWA (contains
            MAILBOX TABLE)

    3.  Point to MAILBOX TABLE entry (address saved in
            EVENT-BLOCK)

    4.  If entry not for this mailbox
        set a 'system-dependent error' STATUS-CODE
        LINK to ERRPRO to log error
        Release use of mailbox
        RETURN to caller

    5.  If previous RECEIVE was returned (implied cancel)
        remove the first message from the mailbox

D.  PRIME EVENT-BLOCK

    1.  Set EVENT-NUMBER to that passed,
        OUTSTANDING-EVENT to 01

E.  RETURN CONTROL TO CALLER

    1.  Set 'successful completion' STATUS-CODE

    2.  Release use of mailbox

    3.  RETURN to caller
```

3.4 Get Message

The GETMSG primitive routine is in program OSIIPC.

```
CALL 'GETMSG' USING IMPUT-MAILBOX-NAME
                    BUFFER
                    BUFFER-SIZE
                    NUMBER-OF-BYTES
                    EVENT-BLOCK-nn
                    STATUS.
```

FUNCTIONS PERFORMED:

```
A.  VALIDATE PARAMETERS

    1.  If buffer-size is zero
        set 'buffer size zero' STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

    2.  If buffer-size is not numeric or greater than max
        set 'buffer size greater than maximum' STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

B.  CHECK EVENT-BLOCK

    1.  If mailbox-name doesn't match that in EVENT-BLOCK
        set 'invalid event block for mailbox named'
            STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

    2.  If EVENT-TYPE is not 01 (RECEIVE)
        set 'not a receive event block' STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

    3.  If EVENT-OUTSTANDING is zero (no event
            outstanding)
        set 'no receive outstanding' STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

C.  READ FIRST MESSAGE FROM MAILBOX

    1.  Serialize on mailbox

    2.  Find mailbox
            address of MAILBOX TABLE entry in EVENT-BLOCK
            address of mailbox in MAILBOX TABLE entry

    3.  If current byte-count equals zero
        set 'receive not satisfied' STATUS-CODE
        LINK to ERRPRO to log error
        release use of mailbox
        RETURN to caller
        4.  If first message is longer than buffer-size
            set 'buffer too small' STATUS-CODE
            LINK to ERRPRO to log error
            clear buffer to blanks

            else        (data is lost)

            move first message to caller's buffer
            set number-of-bytes to length of message
            set 'successful completion' STATUS-CODE

        5.  Remove message read from mailbox
            subtract message length from current-byte-count
            adjust displacements

    D.  CLEAR OUTSTANDING RECEIVE

        1.  Set OUTSTANDING-EVENT to 00

        2.  Clear ECB in EVENT-BLOCK

    E.  RETURN CALL TO CALLER

        1.  Release use of mailbox

        2.  RETURN to caller
```

3.5 <u>Delete Mailbox</u>

The DELMBX primitive routine is in program OSIIPC.

```
CALL 'DELMBX' USING  INPUT-MAILBOX-NAME
                     EVENT-BLOCK-nn
                     STATUS.
```

FUNCTIONS PERFORMED:

```
    A.  CHECK EVENT-BLOCK

        1.  If mailbox-name doesn't match that in EVENT-BLOCK
            set 'invalid event block for mailbox named'
                STATUS-CODE
            LINK to ERRPRO to log error
            RETURN to caller

        2.  If EVENT-TYPE is not 01 (RECEIVE)
            set 'not a receive event block' STATUS-CODE
            LINK to ERRPRO to log error
            RETURN to caller

    B.  DELETE MAILBOX

        1.  Serialize on mailbox and MAILBOX TABLE

        2.  Find mailbox entry in MAILBOX TABLE

        3.  Remove MAILBOX TABLE entry
```

4. FREEMAIN mailbox storage

C. RELEASE THE EVENT-BLOCK

1. Re-initialize the EVENT-BLOCK to character 0's

D. RETURN CONTROL TO CALLER

1. Set 'successful completion' STATUS-CODE

2. Release control of mailbox and MAILBOX CODE

3. RETURN to caller

3.6 Release Event Block

The RELEVB primitive routine is in program OSIIPC

```
CALL 'RELEVB' USING MAILBOX-NAME    (not used in IBM Interface)
                    EVENT-BLOCK
                    STATUS.
```

FUNCTIONS PERFORMED:

A. RELEASE THE EVENT-BLOCK

1. Re-initialize the EVENT-BLOCK to character 0's

B. RETURN CONTROL TO CALLER

1. Set 'successful completion' STATUS-CODE

2. RETURN to caller

3.7 Set Timer

The SETTIM primitive routine is in program OSIIPC.

```
CALL 'SETTIM' USING TIME-INTERVAL
                    EVENT-NUMBER
                    EVENT-BLOCK-nn
                    STATUS.
```

FUNCTIONS PERFORMED:

A. VALIDATE PARAMETERS

```
1. If event-number is zero
   set 'event number zero' STATUS-CODE
   LINK to ERRPRO to log error
   RETURN to caller
```

```
        2.  If event-number is not numeric or is greater than
                22
            set 'event number greater than maximum'
                STATUS-CODE
            LINK to ERRPRO to log error
            RETURN to caller

        3.  If interval requested is zero
            set 'time interval zero' STATUS-CODE
            LINK to ERRPRO to log error
            RETURN to caller

        4.  If interval requested is not numeric or is greater
                than 235959
            set 'time interval greater than maximum'
                STATUS-CODE
            LINK to ERRPRO to log error
            RETURN to caller

    B.  CHECK EVENT-BLOCK

        1.  If EVENT-BLOCK is not all zeros
            set 'event block not initialized' STATUS-CODE
            LINK to ERRPRO to log error
            RETURN to caller

    C.  PRIME EVENT-BLOCK AND OTHER CONTROL AREAS

        1.  Set EVENT-NUMBER to that specified,
                EVENT-TYPE to 02 (TIMER)
                EVENT-OUTSTANDING to 01

        2.  Store address of Timer EVENT-BLOCK in TWA

        3.  Calculate actual expiration time and store in TWA

        4.  Store address of timer ECB (in TASK LIST TABLE) in
                TWA

    D.  RETURN CONTROL TO CALLER

        1.  Set 'successful completion' STATUS-CODE

        2.  RETURN to caller
```

3.8 Cancel Timer

The CNLTIM primitive routine is in program OSIIPC.

```
CALL 'CNLTIM' USING EVENT-BLOCK-nn
                    STATUS.
```

FUNCTIONS PERFORMED:

```
    A.  VALIDATE PARAMETERS

        1.  If EVENT-TYPE is not 02 (timer)
            set 'not a timer event block' STATUS-CODE
            LINK to ERRPRO to log error
            RETURN to caller

    B.  CANCEL TIMER REQUEST

        1.  Clear the TWA timer-related fields

        2.  Re-initialize the EVENT-BLOCK (character 0's)

    C.  RETURN CONTROL TO CALLER

        1.  Set 'successful completion' STATUS-CODE

        2.  RETURN to caller
```

3.9 Wait for an Event

The WAITnn primitive routine is in program OSIIPC.

```
CALL 'WAITnn' USING EVENT-NUMBER
                    STATUS
                    NUMBER-OF-EVENT-BLOCKS
                    EVENT-BLOCK-nn
                    ...
                    EVENT-BLOCK-mm.
```

FUNCTIONS PERFORMED:

```
    A.  VALIDATE PARAMETERS

        1.  If number-of-event-blocks is zero
            set 'number of event blocks zero' STATUS-CODE
            LINK to ERRPRO to log error
            RETURN to caller

        2.  If number-of-event-blocks is not numeric or is
                greater than 22
            set 'number of event blocks greater than max'
                STATUS-CODE
            LINK to ERRPRO to log error
            RETURN to caller

    B.  FIND THE ACTIVE EVENT-BLOCKS (THOSE WITH OUTSTANDING
        EVENTS)

        1.  If fewer EVENT-BLOCKs are passed than
                NUMBER-OF-EVENT-BLOCKS
            set 'fewer event blocks passed than count'
                STATUS-CODE
            LINK to ERRPRO to log error
            RETURN to caller
```

```
2.  If no active EVENT-BLOCKs
    set 'no requests outstanding' STATUS-CODE
    LINK to ERRPRO to log error
    RETURN to caller

3.  If any duplicate EVENT-NUMBERs among the active
        EVENT-BLOCKs
    set 'event numbers not unique' STATUS-CODE
    LINK to ERRPRO to log error
    RETURN to caller
```

C. CHECK EACH ACTIVE EVENT-BLOCK FOR ALREADY-COMPLETED EVENTS

```
1.  TIMER event:
        check expiration time against current time
        if elapsed
          If EVENT-NUMBER is lowest of those completed
          save EVENT-NUMBER and address of EVENT-BLOCK
        else
          issue STIMER REAL with exit routine,
             STIMEXIT, which posts timer ECB in
             EVENT-BLOCK (the exit finds the ECB by
             searching the TASK LIST TABLE by TCB
             addr)
          add timer ECB addr to WAIT list

2.  RECEIVE events:
        serialize on mailbox
        find mailbox
          address of MAILBOX TABLE entry in
             EVENT-BLOCK
          address of mailbox storage in MAILBOX TABLE
             entry
        if any messages in mailbox,
          POST the ECB in the EVENT-BLOCK
          if EVENT-NUMBER is lowest of those completed
             save EVENT-NUMBER and address of
           EVENT-BLOCK
        else
          clear ECB
          add ECB address to WAIT list
        release use of mailbox

3.  LAN RECEIVE events:
        check ECB in the EVENT-BLOCK
        if POSTed
           if EVENT-NUMBER is lowest of those
           completed
             save EVENT-NUMBER and address of
           EVENT-BLOCK
        else
           add ECB address to WAIT list
```

D. IF NO EVENTS HAVE COMPLETED,
WAIT ON THE LIST OF UNCOMPLETED EVENTS

1. WAIT on the list of ECB's until control is returned (at least one event has completed)

E. OF THE EVENTS THAT HAVE COMPLETED,
RETURN THE HIGHEST PRIORITY EVENT

1. Search the ECB list for completed events

2. If EVENT-NUMBER is lowest of those completed save EVENT-NUMBER and address of EVENT-BLOCK

3. Set caller's EVENT-NUMBER to that saved (highest priority)

4. If the returned event is a timer, re-initialize the EVENT-BLOCK (character 0's) clear TWA timer-related fields

F. RETURN CONTROL TO CALLER

1. Set 'successful completion' STATUS-CODE

2. RETURN to caller

3.10 Log an Error Message

The ERRPRO primitive routine is in program OSIERR. This program can be called from an ICAM/IISS program directly or be LINKed to/from the IPC, PRC or IHC programs. It formats and time-stamps the error message and sends it to the mailbox for the Error Log Write task.

```
CALL 'ERRPRO' USING MESSAGE-NUMBER
                    PROGRAM-NAME
                    MESSAGE-TEXT.
```

FUNCTIONS PERFORMED:

A. BUILD ERROR MESSAGE

1. Concatenate parameters into one message

B. SEND ERROR MESSAGE TO ERROR LOG WRITE TASK

1. Send message to mailbox ERRMBX

2. If any error, ignore it

C. RETURN CONTROL TO CALLER

1. RETURN to caller

3.11 Terminate Processing

The ENDRUN primitive routine is in program OSIIBM.

CALL 'ENDRUN'.

STATUS CODES:
none (control is not returned)

FUNCTIONS PERFORMED:

A. TERMINATE THE CALLER'S PROGRAM

1. Restore the ATTACH stub's registers and RETURN

SECTION 4

PROCESS CONTROL PRIMITIVES (PRC)

4.1 Create Process

The CRTPRC primitive routine is in program OSIPRC.

```
CALL 'CRTPRC' USING AP-NAME
                    PROCESS-NAME
                    PRIORITY
                    TYPE-FLAG
                    NTM-STATUS-CODE
                    OPSY-RETCODE
```

FUNCTIONS PERFORMED:

```
A.  VALIDATE PARAMETERS

        1.  If ap-name contains embedded blanks
            set 'invalid application name' in NTM-RETURN
            LINK to ERRPRO to log error
            RETURN to caller

        2.  If process-name contains embedded blanks
            set 'invalid process name' in NTM-RETURN
            LINK to ERRPRO to log error
            RETURN to caller

        3.  If type-flag not valid
            set 'invalid priority' in NTM-RETURN
            LINK to ERRPRO to log error
            RETURN to caller

        4.  If priority not valid *not done now, NTM
                                   doesn't use it*
            set 'invalid priority' in NTM-RETURN
            LINK to ERRPRO to log error
            RETURN to caller

    B.  CHECK FOR DUPLICATE PROCESS-NAME

        1.  Load address of COMMON TABLE from TWA
            (contains TASK LIST TABLE and PROGRAM
            RE-ROUTING TABLE)

        2.  Serialize on the TASK LIST TABLE by
            process-name

        3.  Search TASK LIST TABLE by process-name

        4.  If duplicate process-name
            set 'duplicate process name' in NTM-RETURN
            LINK to ERRPRO to log error
            release control of TASK LIST TABLE
            RETURN to caller
```

```
C.  DETERMINE IF REQUEST SHOULD BE RE-ROUTED

    1.  Search PROGRAM RE-ROUTING TABLE using ap-name
        length of compare is in table entry,
        so the search can be generic

D.  IF ENTRY FOUND, RE-ROUTE CREATE PROCESS REQUEST
    *** not implemented ***

    1.  Format create-process message:
             'CRTPRC PARMS@____,ECB@____'

    2.  Release control of the TASK LIST TABLE

    3.  Send message to mailbox specified in entry

    4.  WAIT on RE-ROUTE ECB

    5.  If POSTed RE-ROUTE ECB not zeros
        set 'create process failed' in NTM-RETURN
        LINK to ERRPRO to log error

        else

        set 'successful completion' in NTM-RETURN

    6.  RETURN to caller

E.  IF ENTRY NOT FOUND, CREATE THE PROCESS

    1.  Search TASK LIST TABLE for an empty slot
           If not empty slot is found,
              GETMAIN additional storage, chain and
              format it
              LINK to ERRPRO to record the overflow

    2.  Format TASK LIST TABLE entry
           move in process-name, ap-name, task
              creation time
           clear end-of-task ECB

    3.  Create the process
           ATTACH EP=(ATTACH-STUB)
              PARAM=(parameter list),
                    (passed to Create Process)
              ECB=(END-OF-TASK ECB),
                    (in TASK LIST TABLE entry)
              dispatching priority = -1, (default)
              ETXR=(END-OF-TASK exit routine address)

F.  RETURN TO CALLER

    1.  Set 'successful completion' in NTM-RETURN

    2.  Release control of the TASK LIST TABLE

    3.  RETURN to caller
```

4.2 Delete Process

The DELPRC primitive routine is in program OSIPRC.

```
CALL 'DELPRC' USING PROCESS-NAME
                    NTM-STATUS-CODE
                    OPSYS-RETCODE.
```

FUNCTIONS PERFORMED:

```
A.  VALIDATE PARAMETERS

    1.  If process-name contains embedded blanks
        set 'invalid process name' in NTM-RETURN
        LINK to ERRPRO to log error
        RETURN to caller

B.  DETERMINE IF REQUEST SHOULD BE REROUTED

    1.  Load address of COMMON TABLE from TWA
            (contains PROGRAM RE-ROUTING TABLE)

    2.  Search PROGRAM RE-ROUTING TABLE using process-name
        ***note that this routine assumes that ap-name***
        ***is the same as the first part of
           process-name***
            length of compare is in table entry,
            so the search can be generic

C.  IF ENTRY FOUND, RE-ROUTE DELETE PROCESS REQUEST
    *** not implemeted ***

    1.  Format delete-process message:
            'DELPRC PARM@____,ECB@____'

    2.  Send message to mailbox specified in entry

    3.  WAIT on RE-ROUTE ECB not zeros

    4.  If POSTed RE-ROUTE ECB not zeros
        set 'delete process failed' in NTM-RETURN
        LINK to ERRPRO to log error

        else

        set 'successful completion' in NTM-RETURN

    5.  RETURN to caller

D.  IF ENTRY NOT FOUND, DELETE THE PROCESS

    1.  Search TASK LIST TABLE for process-name

    2.  If process-name not found (the process has already
            ended)

    3.  Pick up TCB address of process (from TASK LIST
            TABLE entry)
```

4. Issue DETACH TCB=(TCB address)

5. LINK to the End-of-Task exit

E. RETURN TO CALLER

1. Set 'successful completion' in NTM-RETURN

2. RETURN to caller

4.3 Request Process Name

The GETNAM primitive routine is in program OSIPRC.

```
CALL 'REQPRC' USING PROCESS-NAME
                    RETURN CODE
                    NTM-STATUS-CODE.
```

FUNCTIONS PERFORMED:

A. RETURN PROCESS-NAME

1. Pick up our process-name from TWA

2. Move it to PROCESS-NAME

B. RETURN TO CALLER

1. RETURN to caller

SECTION 5

PRC SUPPORT ROUTINES

5.1 ATTACH Stub (Program OSIATCH)

Control will be passed to the ATTACH stub when any task is ATTACHed as a result of a Create Process request.

```
ATTACH 'ATTACH-Stub' PARAM=   AP-NAME
                              PROCESS-NAME
                              PRIORITY
                              TYPE-FLAG
                              NTM-STATUS-CODE
                              OPSYS-RETCODE.
```

FUNCTIONS PERFORMED:

A. OBTAIN TASK WORK AREA

1. GETMAIN Task Work Area (TWA) which consists of Register Save Areas (for OSIIBM and the ATTACH stub) and the Task Work Area for the primitives

2. Clear TWA and prime the TWA indicator field

B. PASS CONTROL TO THE APPLICATION PROGRAM

1. LINK to AP-NAME program passing
   address of original parameter list in register 1
   address of ATTACH Stub's Register Save Area (in the TWA) in register 13

2. When control is returned, RETURN to caller (MVS)

5.2 End-of-task Exit (Program OSIETXR)

Control will be passed to the ETXR routine by MVS when a subtask created by an ATTACH macro completes (normally or abnormally), or when DELPRC LINKs to it after DETACHing a subtask.

FUNCTIONS PERFORMED:

A. FIND AND CHECK THE TASK LIST TABLE ENTRY

1. LOAD COMMON TABLE (contains the TASK LIST TABLE and the MAILBOX TABLE)

2. Serialize on TASK LIST TABLE

3. Search TASK LIST TABLE by TCB address

4. Check return code or completion code in END-OF-TASK ECB in the TASK LIST TABLE entry

5. If return code not zeros
   LINK to ERRPRO to log abend

B. FIND AND REMOVE ANY TABLE ENTRIES

   1. Serialize on MAILBOX TABLE

   2. Search MAILBOX TABLE by TCB address for mailbox(es) and remove table entries

   3. Release control of MAILBOX TABLE

   4. Remove TASK LIST TABLE

   5. Release control of TASK LIST TABLE

C. REMOVE SUBTASK FROM SYSTEM

   1. If subtask was not DETACHed by DELPRC, DETACH subtask

D. RETURN TO CALLER

   1. RETURN to caller (MVS)

SECTION 6

ENVIRONMENT CONTROL MODULES

6.1 MVS Initialization (Program OSIMVSI)

This program is the first program executed when the region comes up (it is the program specified in the EXEC card of the JCL, or as the program to be executed under TSO TEST).

It LOADs the COMMON TABLE, and then searches the MODULE LOAD TABLE and LOADs any modules specified, and searches the TASK ATTACH TABLE and calls CRTPRC to ATTACH any tasks specified, and then searches the PROGRAM LINK TABLE and LINKs to any programs specified. One program that must be in the PROGRAM LINK TABLE is the NTM Monitor AP.

The reason this program issues LOADs of various modules, such as the END-OF-TASK EXIT routine and the VTAM CONTROL BLOCKS, is to insure that these modules are always in virtual storage by keeping their use counts non-zero.

The programs that are ATTACHed are various independent processes, such as the ERROR LOG WRITE task.

NOTE: Since this facility is table-driven, other modules or independent tasks could easily be added in the future --such as a task that submits a batch job when a Create Process request is re-routed to it. The section of this document on the COMMON TABLE contains a description of the tables that drive this program.

FUNCTIONS PERFORMED:

A. INITIALIZATION

1. GETMAIN Task Work Area (TWA) which consists of Register Save Areas (for OSIIBM and OSIMVSI) and the Task Work Area for the primitives

2. Clear TWA and prime the TWA indicator field

3. LOAD COMMON TABLE
   contains: MODULE LOAD TABLE
   TASK ATTACH TABLE
   PROGRAM LINK TABLE
   TASK LIST TABLE

B. LOAD OTHER MODULES

1. Search MODULE LOAD TABLE

   NOTE: This table will contain such modules as: ETXR routine (end-of-task exit) VTAM Control Block Module(s)

2. For any entries found,
   LOAD 'module-name'

C. ATTACH OTHER TASKS

1. Search TASK ATTACH TABLE

   NOTE: This table will contain such tasks as:
   Error Log Write Task

2. For any entries found,
   call CRTPRC to create the task, passing the parameter list in the table entry

D. LINK TO OTHER PROGRAMS

1. Search PROGRAM LINK TABLE

   NOTE: This table will contain such programs as:
   NTM Monitor AP

2. For any entries found,

```
        LINK 'AP-program-name',
                    PARAM=(param list)
                           (same as in Create Process)
```

E. RETURN TO CALLER

1. When control is returned to this program,

```
        (last program LINKed to having returned)
        RETURN to caller (MVS)
                    (IISS system terminates)
```

6.2 Common Table

Description:

The COMMON TABLE is a load module generated from assembler macro calls, which is used to hold information required by all tasks in the system. The initial contents of this table will be set up by OSI but can easily be changed in the future if any changes are desired or new facilities are added.

NOTE: The COMMON TABLE must be link-edited as re-entrant, so that all tasks will reference the same copy of the table. To mark it re-entrant, add the RENT option to the PARM of the link-edit job step.

The COMMON TABLE actually consists of several tables:

1. MODULE LOAD TABLE
2. TASK ATTACH TABLE
3. PROGRAM LINK TABLE
4. MAILBOX TABLE TABLE
5. TASK LIST TABLE
6. PROGRAM RE-ROUTING TABLE

The first three tables are used by the MVS INITIALIZATION program to initialize the region before control is given to the NTM Monitor AP.

1. The MODULE LOAD TABLE contains entries for those load modules and subroutines which should be in virtual storage at all times. By issuing a LOAD for these modules, and then never deleting the modules, the use counts of the modules will always be above zero, thereby insuring that they will be kept in virtual storage.

   Sample macro calls are:

   ```
   COMTBL TYPE=LOAD,MODULE=OSIETXR
   COMTBL TYPE=LOAD,MODULE=xxxx (xxxx = VAX
        port-name)
   ```

2. The TASK ATTACH TABLE contains entries for those independent tasks that need to be started when the region is first brought up. For now, the only entry in this table will be for the Error Log Write Task, but it might be desirable in the future to have a task which submitted a batch job when a Create Process request was re-routed to it.

   Sample macrocalls are:

   ```
   COMTBL TYPE=ATTACH,APNAME=ERLOG,PRNAME=ERLOG1,
          PRIORITY=10,TYPEFLG=1
   COMTBL TYPE=ATTACH,APNAME=SUBMT,PRNAME=SUBMT1,
          PRIORITY=20,TYPEFLG=1
   ```

   A TYPE=ATTACH macro call will result in a table entry containing a parameter list used by OSIMVSI in a CRTPRC call to create the subtask requested.

3. The PROGRAM LINK TABLE contains entries for those programs which should be LINKed to by the MVS INITIALIZATION program. This list could include various initialization and termination programs as well as the NTM Monitor AP itself, although for now the NTM Monitor AP is the only entry.

   Sample macro calls are:

   ```
   COMTBL TYPE=LINK,APNAME=SETUP,PRNAME=SETUP1
   COMBTL TYPE=LINK,APNAME=NTMAP,PRNAME=NTMAP1
   COMBTL TYPE=LINK,APNAME=TRMNMT,PRNAME=TRMNMT1
   ```

   A TYPE=LINK macro call will result in a parameter list used in a LINK to the APNAME program issued by OSIMVSI, passing a parameter list, the same as that passed for Create Process.

4. The MAILBOX TABLE is used to identify and locate every mailbox in the system. An entry in this table is allocated when the mailbox is deleted.

An entry in this table consists of:

the mailbox name,
the address of the storage for the mailbox,
the address of the TCB of the mailbox owner, and
the address of the ECB to POST when a message is sent to the mailbox (in the EVENT-BLOCK).

The number of MAILBOX TABLE entries to allow for is determined by the expected activity of the system and is specified as a parameter in the TYPE-INITIAL macro used to assemble the COMMON TABLE. Provision is made to allocate additional storage for table entries in case the table gets filled, and the number of additional entries is also specified in the macro.

A sample macro call is:

```
COMTBLE TYPE=INITIAL,MBXS=150,MBX=70
```

5. The TASK LIST TABLE is used to keep track of every task executing in the system. An entry in the table is allocated when a process is created and deleted when the task ends normally or is deleted by a Delete Process request.

   An entry in this table consists of:

   the process name,
   the application name,
   the type flag,
   the address of the TCB for the task,
   an ECB that MVS POSTs when the task completes, and
   the task creation time.

   The number of entries to allow for is specified as a parameter in the TYPE=INITIAL macro for the COMMON TABLE. As with the MAILBOX TABLE, provision is made to allocate additional storage for table entries in case the table gets filled, and the number of additional entries is also specified in the macro.

   A sample macro call is:
   ```
   COMTBL TYPE=INITIAL,TSKS=100,TSKI=50
   ```

6. The PROGRAM RE-ROUTING TABLE could be used for two purposes:

   a. To reroute Create Process requests to a task that causes the process to be started in another environment, such as to a task that submits batch jobs, and

   b. To control the subtask hierarchy in the native VTAM region by re-routing a Create Process request to a task that will do the ATTACH on behalf of the caller, rather than attaching the subtask directly to the originating task.

The program name supplied can be the full program name, or a generic name, of the length implied. This would allow, for example, to re-route to batch all query processors beginning with QP.

A sample macro call is:

```
    COMTBL TYPE=REROUTE,PROGRAM=COMPU,TO=MVSMBX
    COMBTL TYPE=REROUTE,PROGRAM=QP,TO=BTCHMBX
```

NOTE: The code will be put into the Create Process primitive to search this table and to be able to re-route the request, but for now no tasks will exist to accept the re-routed request, since neither of the above functions is currently planned for.

Sample source input to generate the COMMON TABLE looks like this:

```
    COMBTL TYPE=INITIAL,MBXS=150,MABXI=70,TAKS=100,
      TSKI=50
    COMTBL TYPE=LOAD,MODULE=OSIETXR
    COMTBL TYPE=LOAD,MODULE=xxxx
      (xxxx = VAX port name)
    COMTBL TYPE=ATTACH,APNAME=ERLOG,PRNAME=ERLOG1
           PRIORITY=10,TYPEFLAG=1
    COMTBL TYPE=LINK,APNAME=NTMAP,PRNAME=NTMAP1
    COMBTL TYPE=REROUTE,PROGRAM=QP,TO=BTCHMBX
    COMBTL TYPE=FINAL
```

SECTION 7

INTER-HOST PRIMITIVES (IHC)

7.1 Initialize Communication with the VAX or Honeywell Level 6

The INILAN primitive routine is in program OSIIHC.

```
CALL 'INILAN' USING PORT-NAME
                    RCV-BLOCK
                    XMIT-BLOCK
                    EVENT-BLOCK-nn
                    STATUS.
```

FUNCTIONS PERFORMED:

```
A.  VALIDATE PARAMETERS

    1.  If PORT-NAME is invalid
        set a 'system-dependent error' STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

    2.  If EVENT-BLOCK is not all zeros
        set 'event block no initialized' STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

B.  CHECK IF COMMUNICATION ALREADY ESTABLISHED

    1.  LOAD VTAM control block load module (name =
        PORT-NAME)

    2.  If already initialized
        (allow a second initialize to be issued)
        a. CLSDST netname (obtained from VTAM c.b.
           module)

C.  PRIME CONTROL BLOCKS

    1.  Move zeros to RCV-BLOCK and XMIT-BLOCK

    2.  Move PORT-NAME to RCV-BLOCK and XMIT-BLOCK

    3.  Move PORT-NAME to EVENT-BLOCK and TWA

    4.  Set address of EVENT-BLOCK in TWA

D.  ESTABLISH CONTROL OF PORT-NAME

    1.  OPEN ACB for this APPLID
        OPNDST netname

    2.  If any error
        set a 'system-dependent error' STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller
```

```
E.  RETURN TO CALLER

    1.  Set 'successful completion' STATUS-CODE

    2.  RETURN to caller
```

7.2 Transmit a Message to the VAX or Honeywell Level 6

The XMTLAN primitive routine is in program OSIIHC.

```
CALL 'XMTLAN' USING XMIT-BLOCK
                    EVENT-BLOCK-nn
                    STATUS.

FUNCTIONS PERFORMED:

A.  VALIDATE PARAMETERS

    1.  If number-of-bytes is zero
          set 'number of bytes zero' STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

    2.  If number-of-bytes is not numeric or is
        greater than the max
        set 'number of bytes greater than maximum'
          STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

B.  CHECK EVENT-BLOCK AND OTHER CONTROL BLOCKS

    1.  If PORT-NAME (in TWA) not in XMIT-BLOCK and
          EVENT-BLOCK
        set 'invalid event block for LAN'
          STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

    2.  If receive is outstanding (EVENT-OUTSTANDING
          is 01)
        set 'receive from LAN is outstanding'
          STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

C.  TRANSMIT THE MESSAGE

    1.  SEND the message

    2.  If any error,
        set a 'system-dependent error' STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

D.  RETURN CONTROL TO CALLER

    1.  Set 'successful completion' STATUS-CODE
```

2. RETURN to caller

7.3 Receive a Message from the VAX or Honeywell Level 6

The RCVLAN primitive routine is in program osiihc.

```
CALL 'RCVLAN' USING RCV-BLOCK
                    EVENT-NUMBER
                    EVENT-BLOCK-NN
                    STATUS.
```

FUNCTIONS PERFORMED:

```
A.  VALIDATE PARAMETERS

    1.  If buffer-size is zero
        set 'buffer size zero' STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

    2.  If buffer-size is not numeric or is greater
          than max
        set 'buffer size greater than maximum'
          STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

    3.  If event-number is zero
        set 'event number zero' STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

    4.  If event-number is not numeric or is greater
          than max
        set 'event-number greater than maximum"
          STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

B.  CHECK EVENT-BLOCK AND OTHER CONTROL BLOCKS

    1:  If PORT-NAME (in TWA) not in RCV-BLOCK and
          EVENT-BLOCK
        set 'invalid event block for LAN' STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

    2.  If RECEIVE is outstanding already
          (EVENT-OUTSTANDING is 01)
        set 'only one outstanding receive permitted'
          STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

    3.  If previous RECEIVE was returned
          (EVENT-OUTSTANDING is 02)
        set OUTSTANDING-EVENT to 00
        Clear ECB          (COMM doesn't want data)
```

```
C.   PRIME EVENT-BLOCK AND OTHER REQUIRED AREAS

     1.   Set EVENT-TYPE = 03
              EVENT-NUMBER = event-number
              OUTSTANDING-EVENT = 01
              address of buffer, buffer length

D.   ISSUE RECEIVE

     1.   RECEIVE, asynchronous

     2.   If any error
          set a 'system-dependent error' STATUS-CODE
          LINK to ERRPRO to log erro
          RETURN to caller

E.   RETURN CONTROL TO CALLER

     1.   Set 'successful completion' STATUS-CODE

     2.   RETURN to caller
```

7.4 <u>Get a Message from the VAX or Honeywell Level 6</u>

The GETLAN primitive routine is in program OSIIHC.

```
CALL 'GETLAN' USING RCV-BLOCK
                    EVENT-BLOCK-nn
                    STATUS.
```

FUNCTIONS PERFORMED:

```
A.   VALIDATE PARAMETERS

     1.   If buffer size is zero
          set 'buffer size zero' STATUS-CODE
          LINK to ERRPRO to log error
          RETURN to caller

     2.   If buffer-size is not numeric or is greater
            than max
          set 'buffer size greater than maximum'
            STATUS-CODE
          LINK to ERRPRO to log error
          RETURN to caller

B.   CHECK EVENT-BLOCK

     1.   If PORT-NAME (in TWA) not in RCV-BLOCK and
            EVENT-BLOCK
          set 'invalid event block for LAN' STATUS-CODE
          LINK to ERRPRO to log error
          RETURN to caller

     2.   If EVENT-TYPE is not RECEIVE from LAN (03)
          set 'not a receive terminal block' STATUS-CODE
          LINK to ERRPRO to log error
          RETURN to caller
```

```
      3.  If no RECEIVE from LAN is outstanding
            (EVENT-OUTSTANDING = 00)
          set 'no receive outstanding' STATUS-CODE
          LINK to ERRPRO to log error
          RETURN to caller

  C.  GET MESSAGE FROM TERMINAL

      1.  If ECB (in EVENT-BLOCK) is not POSTed (by
            VTAM)
          set 'receive not satisfied' STATUS-CODE
          RETURN

      2.  If data length exceeds buffer-size
          set 'buffer too small' STATUS-CODE
            (data is lost)
          LINK to ERRPRO to log error

          else

          move data to buffer
          set NUMBER-OF-BYTES
          set 'successful completion' STATUS-CODE

  D.  CLEAR OUTSTANDING RECEIVE FROM LAN

      1.  Set OUTSTANDING-EVENT to 00

      2.  Clear ECB

  E.  RETURN CONTROL TO CALLER

      1.  RETURN to caller
```

7.5 Cancel a Receive from the VAX or Honeywell Level 6

The CNLLAN primitive routine is in program OSIIHC.

```
CALL 'CNLLAN' USING RCV-BLOCK
                    EVENT-BLOCK-nn
                    STATUS.
```

FUNCTIONS PERFORMED:

```
A.  CHECK EVENT-BLOCK

    1.  If PORT-NAME (in TWA) not in RCV-BLOCK and
          EVENT-BLOCK
        set 'invalid event block for LAN' STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

    2.  If EVENT-TYPE not a RECEIVE from LAN
        (EVENT-TYPE = 03)
        set 'not a receive terminal block' STATUS CODE
        LINK to ERRPRO to log error
        RETURN to caller
        3.  If no RECEIVE from LAN is outstanding
            set 'no receive outstanding' STATUS-CODE
            LINK to ERRPRO to log error
            RETURN to caller

    B.  CANCEL RECIEVE FROM LAN

        1.  CANCEL RECEIVE

        2.  Set OUTSTANDING-EVENT to 00

        3.  Clear ECB     (COMM doesn't want data)

    C.  RETURN CONTROL TO CALLER

        1.  Set 'successful completion' STATUS-CODE

        2.  RETURN to caller
```

7.6 Terminate Communication with the VAX or Honeywell Level 6

The TRMLAN primitive routine is in program OSIIHC.

```
CALL 'TRMLAN' USING PORT-NAME
                    RCV-BLOCK
                    XMIT-BLOCK
                    EVENT-BLOCK-nn
                    STATUS.
```

FUNCTIONS PERFORMED:

```
A.  VALIDATE PARAMETERS

    1.  If PORT-NAME is invalid
        set a 'system-dependent error' STATUS-CODE
        LINK to ERRPRO to log error
        RETURN to caller

B.  CLEAR CONTROL BLOCKS

    1.  Move zeros to RCV-BLOCK and XMIT-BLOCK

    2.  Move zeros to EVENT-BLOCK

    3.  Clear TWA fields

C.  TERMINATE CONTROL OF PORT-NAME

    1.  CLSDST netname
        CLOSE ACB for this APPLID

    2.  If any error
        set a 'system-dependent error' STATUS-CODE
        LINK to ERRPRO tolog error
        RETURN to caller
```

D. RETURN CONTROL TO CALLER

1. Set 'successful completion' STATUS-CODE
2. RETURN to caller

SECTION 8

INTERFACE TO CICS APPLICATION

The support for a CICS application such as PIOS is handled by the following software:

1. A Pseudo-AP, which runs as an MVS subtask in the same region as the NTM, and acts like an integrated AP. It calls the VTI to convert data streams to and from 3270 format and also calls the CICS interface to send those data streams to CICS and receive data back from CICS.

2. OSICICS, the interface to CICS, which is the only layer of this software which knows it is dealing with CICS. It is called by the Pseudo-AP to send and receive data streams to and from CICS. It in turn branches to the 3270 emulator program to perform the actual communication with CICS.

   It also generates some messages for CICS, such as a CICS sign-on message, and sign-off message and interprets the output from CICS as to whether or not the requested function has competed successfully.

3. OSI3270, a 3270 emulator program in native VTAM, which logs on to CICS, and sends and receives data to and from CICS as if it were a 3270. This program does not know it is dealing with CICS but merely handles the data transfer to whatever VTAM application it is in communication with.

4. A remote CICS, running the CICS application (PIOS).

8.1 Pseudo-AP

The Pseudo-AP is started by the NTM MPU for the PIOS AP cluster. It is called a Pseudo-AP because it is not the real application. It performs the following functions:

A. INITIALIZE

```
1.  CALL 'INITAL' (NTM service) to create input mailbox

2.  CALL 'GETUSR' (NTM service) to obtain user id

3.  CALL 'INICIS' (CICS interface) USING USER-ID
                                         BUFFER
                                         BUFFER-LENGTH
                                         EVENT-BLOCK-C
                                         STATUS-CICS.
```

Note: Data returned in the buffer will consist of a data stream to clear the screen.

B. SEND MESSAGE TO TERMINAL OPERATOR AND RECEIVE RESPONSE

1. CALL 'VTI???' (VTI 3270 to neutral) USING BUFFER
```
                                             BUFFER-LENGTH
                                             NUMBER-OF-BYTES
                                             STATUS-VTI.
```

Note: Buffer and number-of-bytes will be mcdified to contain the neutral format data stream to be sent to the terminal and its length.

2. CALL 'OIVTI' (VTI output/input) USING BUFFER
```
                                         BUFFER-LENGTH
                                         NUMBER-OF-BYTES
                                         STATUS-VTI.
```

Note: The VTI will send the neutral data stream to the terminal attached to the VAX and wait for input from the operator. When control is returned to this program, the buffer and number-of-bytes will be modified to contain the neutral format data stream received back from the terminal and its length.

3. When response is received from operator,

CALL 'VTI???' (VTI neutral to 3270) USING BUFFER
```
                                          BUFFER-LENGTH
                                          NUMBER-OF-BYTES
                                          STATUS-VTI.
```

Note: Buffer and number-of-bytes will be modified to contain the 3270 format data stream to send to CICS and its length

C. END PROCESS IF OPERATOR INDICATES TO

1. IF ATTENTION-KEY (equivalent of the break key in the VAX) has been struck:

   a. Issue a call to OSICICS to terminate:
```
CALL 'TRMCIS' USING EVENT-BLOCK-C
                    STATUS-CICS.
```

   b. Format a session terminate message for the operator

   c. Convert the data stream to neutral

```
CALL 'VTI???' USING BUFFER
                    BUFFER-LENGTH
                    NUMBER-OF-BYTES
                    STATUS-VTI.
```

d. Send to operator with no response

```
CALL 'OTVTI'  USING BUFFER
                    BUFFER-LENGTH
                    NUMBER-OF-BYTES
                    STATUS-VTI.
```

e. Terminate process

```
CALL 'TRMNAT' (NTM service) using...
(process terminates)
```

D. SEND OPERATOR INPUT TO CICS AND RECEIVE RESPONSE

1.
```
CALL 'SNDCIS' (CICS interface) USING BUFFER
                                     NUMBER-OF-BYTES
                                     EVENT-BLOCK-C
                                     STATUS-CICS.
```
Note: Data will be sent to CICS

2.
```
CALL 'RCVCIS' (CICS interface) USING BUFFER
                                     BUFFER-LENGTH
                                     NUMBER-OF-BYTES
                                     EVENT-BLOCK-C
                                     STATUS-CICS.
```

3.
```
CALL 'GETCIS' (CICS interface) USING BUFFER
                                     BUFFER-LENGTH
                                     NUMBER OF BYTES
                                     EVENT-BLOCK-C
                                     STATUS-CICS.
```

4. If receive is satisfied, branch to step B.1.

E. TERMINATE PROCESS IF SESSION IS LOST

```
1.  If receive is not satisfied,
    a.  Set timer
        CALL 'SETTIM' USING TIME-INTERVAL
                            EVENT-NUMBER
                            EVENT-BLOCK-T
                            EVENT-BLOCK-C
                            STATUS-CODE.

    b.  WAIT for receive or timer to complete
        CALL 'WAIT02' USING EVENT-NUMBER
                            STATUS-CODE
                            NUMBER-OF-EVENT-BLOCKS
                            EVENT-BLOCK-T.

2.  If EVENT-NUMBER indicates timer expired,
    a.  Cancel outstanding RECEIVE
        CALL 'CNLCIS' USING EVENT-BLOCK-C
                            STATUS-CICS.

    b.  Terminate 3270 emulation
        CALL 'TRMCIS' USING BUFFER
                            BUFFER-LENGTH
                            EVENT-BLOCK-C
                            STATUS-CICS.

    c.  Format termination message for terminal operator

    d.  Send message to operator with no response

   CALL 'VTI???' (VTI 3270 to neutral) USING BUFFER
                                             BUFFER-LENGTH
                                             NUMBER-OF-BYTES
                                             STATUS-VTI
   CALL 'OTVTI' (VTI output)  USING BUFFER
                                    BUFFER-LENGTH
                                    NUMBER-OF-BYTES
                                    STATUS-VTI.
   e.  Terminate

   CALL 'TRMNAT' (NTM service) using...
   (process terminates)

3.  If EVENT-NUMBER indicates receive was satisfied,
      CALL 'GETCIS' (CICS interface) USING BUFFER
                                           BUFFER-LENGTH
                                           NUMBER-OF-BYTES
                                           EVENT-BLOCK-C
                                           STATUS-CICS.

4.  Branch to step B.1.
```

SECTION 9

CICS INTERFACE PRIMITIVES

9.1 Initialize Communication with CICS

```
CALL 'INICIS'  USING USER-ID
                     BUFFER
                     BUFFER-LENGTH
                     EVENT-BLOCK
                     STATUS-CICS.
```

FUNCTIONS PERFORMED:

```
A.  VALIDATE PARAMETERS

    1.  If parameter list is invalid
        set 'invalid parameter list specified' STATUS-CICS
        Branch to ERRPRO to log the error
        RETURN to caller

    2.  If user-id is blanks or zeros
        set 'invalid user id' STATUS-CICS
        Branch to ERRPRO to log the error
        RETURN to caller

    3.  If buffer-length is zeros
        set 'buffer length zero' STATUS-CICS
        Branch to ERRPRO to log the error
        RETURN to caller

    4.  If buffer-length is non-numeric or is greater than
        max
        set 'buffer length exceeds maximum' STATUS-CICS
        Branch to ERRPRO to log the error
        RETURN to caller

B.  LOG ON TO CICS

    1.  Prime EVENT-BLOCK for INITIALIZE:
        action code for initialize

    2.  Branch to OSI3270, passing the EVENT-BLOCK

    3.  If status indicates no session is available
        set 'retry later' STATUS-CICS
        Branch to ERRPRO to log the error
        RETURN to caller

    4.  If status indicates a VTAM error
        set 'system-dependent error' STATUS-CICS
        Branch to ERRPRO to log the error
        RETURN to caller

C.  SIGN ON TO CICS

    1.  Build the SIGN-ON message for CICS in the caller's
        buffer using the user-id that was passed
```

```
     2.  Prime EVENT-BLOCK for SEND:
         action code for send
         address of data
         length of data

     3.  Branch to OSI3270, passing the EVENT-BLOCK

     4.  If status indicates a VTAM error
         set 'system-dependent error' STATUS-CICS
         Branch to ERRPRO to log the error
         RETURN to caller

D.  RECEIVE RESPONSE FROM CICS TO SIGN ON

     1.  Prime EVENT-BLOCK for RECEIVE:
         action code for receive
         address of buffer
         length of buffer

     2.  Branch to OSI3270, passing the EVENT-BLOCK

     3.  If status indicates a VTAM error
         set 'system-dependent error' STATUS-CICS
         Branch to ERRPRO to log the error
         RETURN to caller

     4.  Prime EVENT-BLOCK for GET:
         action code for get
         address of buffer
         length of buffer

     5.  Branch to OSI3270, passing the EVENT-BLOCK

     6.  If RECEIVE is not satisfied
         set a timer and wait on the timer and the receive
         RETURN to caller
         (caller should issue a wait followed by another
         GET)

     7.  If the timer expired,
         prime EVENT-BLOCK for TERMINATION
         branch to OSI3270 to terminate communication
         set 'initialization failed' STATUS-CICS
         branch to ERRPRO to log the error
         RETURN to caller

     8.  If the sign on was rejected (look at msg from
         CICS)
         prime EVENT-BLOCK for TERMINATION
         branch to OSI3270 to terminate communication
         set 'initialization failed' STATUS-CICS
         branch to ERRPRO to log the error
         RETURN to caller

         Note: The response sent by CICS to the SIGN-ON is
               discarded.

E. FORMAT RESPONSE FOR OPERATOR
```

```
        1.  Build a CLEAR SCREEN data stream in the caller's
            buffer ERASE WRITE, FREE KEYBOARD, NO DATA

        2.  Set NUMBER-OF-BYTES to the length of the data
            stream

    F.  RETURN CONTROL TO CALLER

        1.  Set a 'successful completion'

        2.  RETURN to caller
```

9.2 Send Message to CICS

```
CALL 'SNDCIS'  USING BUFFER
                     NUMBER-OF-BYTES
                     EVENT-BLOCK
                     STATUS-CICS.
```

FUNCTIONS PERFORMED:

```
    A.  VALIDATE PARAMETERS

        1.  If parameter list is invalid
            set 'invalid parameter list specified' STATUS-CICS
            Branch to ERRPRO to log the error
            RETURN to caller

        2.  If number-of-bytes is zeros
            set 'number of bytes zero' STATUS-CICS
            Branch to ERRPRO to log the error
            RETURN to caller

        3.  If number-of-bytes is non-numeric or is greater
                than max
            set 'number of bytes exceeds maximum' STATUS-CICS
            Branch to ERRPRO to log the error
            RETURN to caller

    B.  SEND MESSAGE TO CICS

        1.  Prime EVENT-BLOCK for SEND:
            action code for send
            address of data
            length of data

        2.  Branch to OSI3270, passing the EVENT-BLOCK

    C.  CHECK RESPONSE

        1.  If status indicates a VTAM error
            set 'system-dependent error' STATUS-CICS
            Branch to ERRPRO to log the error
            RETURN to caller

    D.  RETURN CONTROL TO CALLER

        1.  Set 'successful completion' STATUS-CICS
        2.  RETURN to caller
```

9.3 Receive a Message from CICS

```
CALL 'RCVCIS'  USING BUFFER
                     BUFFER-LENGTH
                     NUMBER-OF-BYTES
                     EVENT-BLOCK
                     STATUS-CICS.

FUNCTIONS PERFORMED:

    A.  VALIDATE PARAMETERS

        1.  If parameter list is invalid
            set 'invalid parameter list specified' STATUS-CICS
            Branch to ERRPRO to log the error
            RETURN to caller

        2.  If buffer-length is zeros
            set 'buffer length zero' STATUS-CICS
            Branch to ERRPRO to log the error
            RETURN to caller

        3.  If buffer-length is non-numeric or is greater than
               max
            set 'buffer length exceeds maximum' STATUS-CICS
            Branch to ERRPRO to log the error
            RETURN to caller

    B.  RECEIVE MESSAGE FROM CICS

        1.  Prime the EVENT-BLOCK for a RECEIVE:
            action code for receive
            address of buffer
            length of buffer

        2.  Branch to OSI3270, passing the EVENT-BLOCK

    C.  CHECK RESPONSE

        1.  If status indicates a VTAM error
            set 'system-dependent error' STATUS-CICS
            Branch to ERRPRO to log the error
            RETURN to caller

    D.  RETURN CONTROL TO CALLER

        1.  Set 'successful completion' STATUS-CICS

        2.  RETURN to caller
```

9.4 Get a Message from CICS

```
CALL 'GETCIS'  USING BUFFER
                     BUFFER-LENGTH
                     NUMBER-OF-BYTES
                     EVENT-BLOCK
                     STATUS-CICS.
```

FUNCTIONS PERFORMED:

```
A.  VALIDATE PARAMETERS

    1.  If parameter list is invalid
        set 'invalid parameter list specified' STATUS-CICS
        Branch to ERRPRO to log the error
        RETURN to caller

    2.  If buffer-length is zeros
        set 'buffer length zero' STATUS-CICS
        Branch to ERRPRO to log the error
        RETURN to caller

    3.  If buffer-length is non-numeric or is greater than
        max
        set 'buffer length exceeds maximum' STATUS-CICS
        Branch to ERRPRO to log the error
        RETURN to caller

B.  GET MESSAGE FROM CICS

    1.  Prime EVENT-BLOCK for GET:
        action code for get
        address of buffer
        length of buffer

    2.  Branch to OSI3270, passing the EVENT-BLOCK

C.  CHECK RESPONSE

    1.  If status indicates a VTAM error
        set 'system-dependent error' STATUS-CICS
        Branch to ERRPRO to log the error
        RETURN to caller

    2.  If receive is not satisfied
        set 'receive not satisfied' in STATUS-CICS
        Branch to ERRPRO to log the error
        RETURN to caller

    3.  Set NUMBER-OF-BYTES to length of message from CICS

D.  RETURN CONTROL TO CALLER

    1.  Set 'successful completion' STATUS-CICS

    2.  RETURN to caller
```

9.5 Cancel Receive from CICS

```
CALL 'CNLCIS'  USING EVENT-BLOCK
                     STATUS-CICS.
```

FUNCTIONS PERFORMED:

```
      A.  VALIDATE PARAMETERS

          1.  If parameter list is invalid
              set 'invalid parameter list specified' STATUS-CICS
              Branch to ERRPRO to log the error
              RETURN to caller

      B.  CANCEL RECEIVE

          1.  Prime EVENT-BLOCK for a CANCEL:
              action code for cancel

          2.  Branch to OSI3270, passing the EVENT-BLOCK

      C.  CHECK RESPONSE

          1.  If status indicates a VTAM error
              set 'system-dependent error' STATUS-CICS
              Branch to ERRPRO to log the error
              RETURN to caller

      D.  RETURN CONTROL TO CALLER

          1.  Set 'successful completion' STATUS-CICS

          2.  RETURN to caller
```

9.6 Terminate Communication with CICS

```
CALL 'TRMCIS'  USING BUFFER
                     BUFFER-LENGTH
                     EVENT-BLOCK
                     STATUS-CICS.
```

FUNCTIONS PERFORMED:

```
      A.  VALIDATE PARAMETERS

          1.  If number of parameters is fewer than two
              set 'invalid parameter list specified' STATUS-CICS
              Branch to ERRPRO to log the error
              RETURN to caller

      B.  SIGN OFF CICS

          1.  Build the SIGN-OFF message for CICS in the
              caller's buffer

          2.  Prime EVENT-BLOCK for a SEND:
              action code for send
              address of data
              length of buffer
      3.  Branch OSI3270, passing the EVENT-BLOCK

      4.  If status indicates a VTAM error
          set 'system-dependent error' STATUS-CICS
          branch to ERRPRO to log the error
          branch to LOG OFF CICS

      5.  Prime EVENT-BLOCK for a RECEIVE:
          action code for receive
          address of buffer
          length of buffer

      6.  Branch to OSI3270, passing the EVENT-BLOCK

      7.  If status indicates a VTAM error
          set 'system-dependent error' STATUS-CICS
          branch to ERRPRO to log the error
          branch to LOG OFF CICS

      8.  Prime EVENT-BLOCK for GET:
          action code for get
          address of buffer
          length of buffer

      9.  Branch to OSI3270, passing the EVENT-BLOCK

     10.  If RECEIVE has not completed
          branch to LOG OFF CICS

          Note:  The response sent by CICS to the SIGN-OFF
                 is discarded.

  C.  LOG OFF CICS AND VTAM

      1.  Prime EVENT-BLOCK for a TERMINATE:
          action code for terminate

      2.  Branch to OSI3270, passing the EVENT-BLOCK

  D.  CHECK RESPONSE

      1.  If status indicates a VTAM error
          set system-dependent error' STATUS-CICS
          Branch to ERRPRO to log the error
          RETURN to caller

  E.  RETURN CONTROL TO CALLER

      1.  Set 'successful completion' STATUS-CICS

      2.  RETURN to caller
```

SECTION 10

3270 EMULATOR PROGRAM

The 3270 emulator program is designed to appear to the CICS as a local 3277 type terminal. All data flow control requests are handled by VTAM. The normal flow send/receive mode is full duplex. FM profile 2 and TS profile 2 are used.

The emulator program is linked to by OSICICS, which has determined which function the 3270 emulator is to perform and has placed the required parameters and the appropriate action code in the EVENT-BLOCK. The address of the EVENT-BLOCK is passed via a parameter list (with only one parameter) in register 1 to OSI3270.

FUNCTIONS PERFORMED:

```
A.  INITIAL LOGON TO CICS

    1.  LOAD VTAM SLU TABLE and search for an available
        entry

    2.  If no entry is available,
        set 'no SLU control blocks available' in
        EVENT-BLOCK
        RETURN to caller
        else mark the SLU TABLE entry in use

    3.  OPEN VTAM ACB

    4.  If OPEN failed,
        release use of SLU TABLE entry in use
        set 'system dependent error' in EVENT-BLOCK
        RETURN to caller

    5.  Issue a SETLOGON - this macro must be issued
        before the 3270 emulator can request a session
        (VTAM restriction)

    6.  If register 15 is not zero
        set 'system dependent error' in EVENT-BLOCK
        move return code and feed back fields into
        EVENT-BLOCK
        RETURN to caller

    7.  Issue a REQSESS - this macro must specify the
        applid of the application that the 3270 emulator
        wishes to access

    8.  If register 15 is not zero
        set 'system dependent error' in EVENT-BLOCK
        move return code and feed back fields into
           EVENT-BLOCK
        RETURN to caller
```

9. A WAIT is then issued which will be posted when the SCIP exit gets control as a result of CICS responding to the REQSESS which was issued previously

10. The SCIP exit performs the following functions:

    a. If a BIND has flowed from CICS

        1. Issue an OPNSEC which acts as a positive response to the BIND

        2. If register 15 is not zero
           set 'system dependent error' in EVENT-BLOCK
           move return code and feed back fields into EVENT-BLOCK RETURN to caller

        3. POST ECB which is being waited on so the mainline program can continue processing

    b. If an UNBIND has flowed from CICS, POST the ECB which the TERMSESS is waiting on

11. Set 'successful completion' in EVENT-BLOCK

12. RETURN to caller

B. SEND TO CICS

1. Issue an asynchronous SEND to CICS

2. WAIT on ECB to be POSTED by VTAM signaling completion of operation

3. If register 15 is not zero
   set 'system dependent error' in EVENT-BLOCK
   move return code and feedback fields into
   EVENT-BLOCK
   RETURN to caller

4. Set 'successful completion' in EVENT-BLOCK

5. RETURN to caller

C. RECEIVE from CICS

1. Issue an asynchronous RECEIVE from CICS

2. If register 15 is not zero
   set 'system dependent error' in EVENT-BLOCK
   move return code and feed back fields into
   EVENT-BLOCK
   RETURN to caller

3. Set 'successful completion' in EVENT-BLOCK

4. RETURN to caller

D. GET data from CICS

```
1.  If RECEIVE ECB has not been posted
    set 'receive not satisfied' in EVENT-BLOCK
    RETURN to caller

2.  Clear RECEIVE ECB

3.  Set 'successful completion' in EVENT-BLOCK

4.  RETURN to caller
```

E. CANCEL an outstanding RECEIVE

```
1.  Issue a RESETSR to cancel outstanding RECEIVE

2.  If register 15 is not zero
    set 'system dependent error' in EVENT-BLOCK
    move return code and feed back fields into
       EVENT-BLOCK
    RETURN to caller

3.  Clear RECEIVE ECB

4.  Set 'successful completion' in EVENT-BLOCK

5.  RETURN to caller
```

F. TERMINATE session with CICS

```
1.  Issue a TERMSESS to terminate session with CICS

2.  WAIT on TERMSESS ECB which will be POSTED by SCIP
    exit

3.  If register 15 is not zero
    set 'system dependent error' in EVENT-BLOCK
    move return code and feed back fields into
       EVENT-BLOCK
    RETURN to caller

4.  CLOSE VTAM ACB

5.  If register 15 is not zero
    set 'system dependent error' in EVENT-BLOCK
    move return code and feed back fields into
       EVENT-BLOCK
    RETURN to caller

6.  Set 'successful completion' in EVENT-BLOCK

7.  RETURN to caller
```

APPENDIX A

MAILBOX LOGIC

BUFFER WRAPAROUND

The algorithm being used for mailboxes uses little cpu time for data movement but is rather complex. A mailbox is a fixed amount of storage, like a buffer, with messages being written to and read from it concurrently. The mailbox routine must keep track of where the first message to be read is and also where to write the next message sent to the mailbox. The problem is complicated by the fact that the messages are variable-length.

The routine keeps adding messages after the messages already in the mailbox until a message being sent doesn't fit. At this point, a check is made to see if the message will fit at the beginning of the mailbox. This will only happen if enough messages have already been read from the mailbox. If the message fits, it is put there, resulting in message sending having wrapped around.

Message reading also wraps around. When a message is to be read, but the spot where it is to be read from points beyond where any data has been put, message reading wraps around to the beginning of the mailbox buffer.

The storage in the mailbox can get fragmented using this technique. A message can be rejected even though there is enough space, because the space is not contiguous, due to the effects of wrapping around.

FUNCTIONS PERFORMED:

A. Sending a Message

```
If current-bytes = zero
set remove-displacement to zero
set insert-displacement to zero
set high-water-mark-displacement to zero
if msg-length <= mbx-size
    move message to <buffer + 0>
    add msg-length to insert-displacement
    add msg-length to current-bytes
    set high-water-mark-displacement to
      insert-displacement
else
    return 'msg too long' error

If insert-displacement > remove-displacement
    If insert-displacement + msg-length <= mbx-size
        move message to (buffer + insert-displacement)
        add msg-length to current-bytes
        set high-water-mark-displacement to
           insert-displacement
    else
        if msg-length <= remove-displacement
           move message to (buffer + 0)
```

```
                set insert-displacement to msg-length
             else
                return 'mbx full' error

    If insert-displacement < remove-displacement
        if msg-length <= (remove-displacement -
                          insert-displacement)
            move message to (buffer + insert-displacement)
            add msg-length to insert-displacement
            add msg-length to current-bytes
        else
            return 'mbx full' error

    If insert-displacement = remove-displacement
        and current-bytes not = zero
        return 'mbx full' error
```

B. Getting a Message

```
    If current-bytes not = zero
        if remove-displacement not equal high-water-mark
            move msg (buffer + remove-displacement) to
               caller's buffer
            add msg-length to remove-displacement
        else
            move message (buffer + 0) to caller's buffer
            set remove-displacement to msg-length
            set high-water-mark to insert-displacement
    else
        return 'receive not satisfied' error
```

rd = remove displacement, id = insert displacement

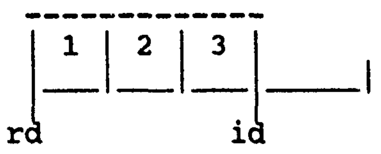

Figure 1. Simple case, no wraparound

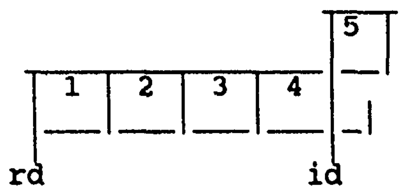

Figure 2. Mailbox full

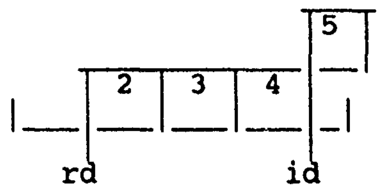

Figure 3. Condition for buffer wraparound on send

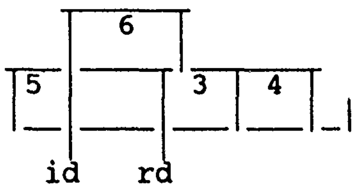

Figure 4. Mailbox full (due to fragmentation)

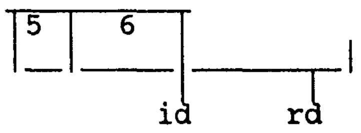

Figure 5. Condition for buffer wraparound on read

APPENDIX B

MODULE STRUCTURE CHART

```
                    1+---------------+           +------+
                 +---| ATTACH STUB   |--GETMAIN--| TWA  |
                 |   +---------------+           +------+
                 |
                LINK
                 |
                 |
                 |
                 |
                 |  2  +--------------+
                 +-->  | COBOL PGM:   |
                       |  COMM, NTM,  |
                       |  QP, or      |
                       |  PSEUDO-AP   |
                    3  +--------------+
                 +-----| OSIBM        |
                 |     +--------------+
                LINK*
                 |
                 |
<----------------------------------------------------------->
|       |       |           |           |                  |
|       |       |           |           |                  |
|       |       |           |           |                  |
| 4 +-------+ 5 +-------+ 6 +-------+ 7 +------+ 8 +-------+
|   | IPC   |   | PRC   |   | IHC   |   | CON  |   | CICS  |
| +-|       | +-|       | +-|       | +-|      | +-| INT.  |
| | +-------+ | +-------+ | +-------+ | +------+ | +-------+
| |         |           |           |          |      |
|LINK*       LINK*       LINK*       LINK*      LINK*  LINK*
| |         |           |           |          |      |
| |         |           |           |          |      |
| |         |           |           |          |  9 +-------+
>->-----------> -----------|-----------<----------<   | 3270  |
                          |                          | EMU   |
                          |                          +-------+
10 +---------+  11 +---------+  12 +----------+
   |  ETXR   |     | ERRPRO  |     | IHCEXT   |
   |         |     |         |     |          |
   +---------+     +---------+     +----------+

13 +-------+ 14 +-------+ 15 +-------+ 16 +-------+ 17 +--------+
   |COMMON |    |VTAM CB|    |VTAM CB|    |VTAM CB|    |VTAM SLU|
   |TABLE  |    |VAX    |    |HL6    |    |CON    |    |TABLE   |
   +-------+    +-------+    +-------+    +-------+    +--------+
```

* These LINKS are to be replaced with direct branches.

MODULES:

1. The ATTACH STUB which GETMAINs and initializes the TWA

2. The COBOL program, such as COMM01 or COMM02, the NTM Monitor, the NTM MPUs, the Query Processors, or the Pseudo-APs

3. OSIIBM, which records the type of call and links to the appropriate interface routine (IPC, PRC, IHC, etc.)

4. IPC, the Inter-Process Primitives interface program

5. PRC, the Process Control Primitives interface program

6. IHC, the Inter-Host Primitives interface program

7. CON, the Console Primitives interface program

8. The CICS Interface, which allows the Pseudo-AP to talk to CICS

9. The 3270 Emulator, which emulates a local 3270 through VTAM

10. ETXR, the End-of-Task exit

11. ERRPRO, which can be linked to by the COBOL program or by any interface program, and formats and sends error messages

12. IHCEXT, the VTAM error exits for IHC

13. The COMMON TABLE, a load module which contains data required to run the IISS Test Bed on IBM

14, 15, 16. VTAM control block load modules for the VAX, the Honeywell Level 6, and the NTM console

17. The VTAM SLU table, which contains multiple sets of VTAM control blocks to use to emulate 3270s into CICS

NOTES:

1. The ATTACH STUB is the first module to get control when a task is created

2. The ATTACH STUB LINKs to the COBOL program

3. CALL statements issued by the COBOL program are resolved by entry points in OSIIBM

4. OSIIBM LINKs to the appropriate interface routine to perform the requested function

5. The interface programs use data and control blocks contained in other load modules such as the Common Table, which are LOADed at system utilization

PROCESSING FLOW

```
    ATTACH STUB (R1 => CRTPRC parmlist)
    +--------------------------------+      TWA
    | GETMAIN TWA                    |---->+------------------------+
    | establish RSA (in TWA)         |     | RSA for ATTACH STUB    |
    | store A(TWA) in TWA            |     | RSA for OSIIBM         |
    | LOAD COMMON TABLE              |     | RSAs for IPC,PRC,IHC,  |
    | store A(COMMON TABLE) in TWA   |     |   etc.                 |
 +--| LINK to COBOL program          |--+  | A(TWA)                 |
 |  | RETURN  (task ends)            |  |  | A(COMMON TABLE)        |
 |  +--------------------------------+  |  | type of primitive call |
LINK                                    |  | A(parmlist) for        |
 |   COBOL pgm (R1 => CRTPAC parmlist)  |  | for primitive          |
 +->+---------------------------------+ |  | ERRPRO parmlist and    |
    |                                 | |  | parms                  |
    |                                 | |  +------------------------+
|---| CALL 'SNDMSG' USING MBX-NAME    | |
|   |                     BUFFERS     | |   COMMON TABLE
|   |                     NUM-BYTES   | +->+------------------------+
|   |                     STATUS.     |    |   .                    |
|   |   .                             |    | MAILBOX TABLE          |
|   +---------------------------------+    | TASKLIST TABLE         |
|   | OSIIBM (linked with COBOL pgm)  |    |   .                    |
|   |   .                             |    +------------------------+
+-> | ENTRY SNDMSG                    |
    | search thru RSAs to find TWA    |
    | set type of call in TWA         |
    | store A(SNDMSG parms) in TWA    |
+-- | LINK to IPC program             |
|   | RETURN to COBOL pgm             |
|   +---------------------------------+
LINK *
|
|    IPC (R1 => A(TWA))
+-> +---------------------------------+
    | Load A(TWA)                     |
    | establish IPC RSA in TWA        |
    | load A(COMMON TABLE) from TWA   |
    | a call to send a message        |
    |   .                             |
    | SNDMSG RTN:                     |
    | load A(SNDMSG parms) from TWA   |
    | search MAILBOX TABLE            |
    | if mailbox not found,           |
+-- |    LINK to ERRPRO               |
|   | RETURN                          |
```

PROCESSING FLOW (Continued)

```
|   +---------------------------------+
LINK *
|
|    ERRPRO (R1 => ERRPRO parmlist (in TWA))
+-> +---------------------------------+
    | format error message            |
    | send message to mailbox ERRMBX  |
    | RETURN                          |
    +---------------------------------+
```

SYSTEM INITIALIZATION

```
     MVS INITIALIZATION
    +----------------------------+      TWA
    | GETMAIN TWA                |---->+-----------------------+
    | establish RSA (in TWA)     |     | RSA for MVS INIT      |
    | store A(TWA) in TWA        |     | RSA for OSIIBM        |
    | LOAD COMMON TABLE          |--+  | RSAs for IPC,PRC,IHC, |
    | store A(COMMON TABLE) in   |  |  |   etc.                |
    |  TWA                       |  |  | A(TWA)                |
    | build own TASKLIST TABLE   |  |  | A(COMMON TABLE)       |
    |  entry                     |  |  |                       |
    | LOAD modules in MODULE     |  |  +-----------------------+
    |  TABLE & save addrs in     |  |
    |  MODULE TABLE: IPC, PRC,   |  |   COMMON TABLE
    |  IHC, CON, ERRPRO, CICS    |  +->+-----------------------+
    |  Interface,3270 Emulator,  |     | MODULE LOAD TABLE     |
    |  VTAM PLU control blocks,  |     | TASK ATTACH TABLE     |
    |  VTAM SLU control block    |     | PROGRAM LINK TABLE    |
    |      table                 |     |   .                   |
    | ATTACH tasks in ATTACH     |     | MAILBOX TABLE         |
    |  TABLE using the CRTPRC    |     | TASKLIST TABLE        |
    |    primitive: ERROR LOG    |--+  +-----------------------+
    |    WRITE task              |  |
 +--|  LINK to programs in LINK  |  |
 |  |  TABLE (e.g., the NTM      |  |
 |  |     Monitor AP)            |  |   ATTACH STUB
 |  |  RETURN (region terminates)|  +->+-----------------------+
 |  +----------------------------+     | GETMAIN TWA, etc.     |
 |  | OSIIBM                     |     | LINK to ERROR LOG     |
 |  +----------------------------+  +--| WRITE                 |
 |                                  |  +-----------------------+
 |                                  LINK
 |                                  |
LINK                                |   ERROR LOG WRITE task
 |                                  +->+-----------------------+
 |                                     | CALL 'CRTMBX' USING   |
 |                                     |           'ERRMBX'    |
 |                                     |            SIZE ...   |
 |                                     |                       |
 |NTM Monitor AP (R1 => CRTPRC parms)  | CALL 'WAIT01' USING   |
 +->+----------------------------+     |            MBX ...    |
                                       |    .                  |
+-- | CALL 'CRTPRC' USING MON-MPU |    +-----------------------+
|   |                    PRIORITY |    | OSIIBM                |
|   |                    STATUS.  |    +-----------------------+
|   |                             |
|   +-----------------------------+
|   |OSIIBM(linked with COBOL pgm |
|   |    .                        |
+-->|ENTRY CRTPRC                 |
    |search thru RSAs to find TWA |
    |set type of call in TWA      |
    |store A(CRTPRC parms) in TWA |
+-- |LINK to PRC program          |
|   |RETURN to COBOL pgm          |
|   +-----------------------------+
LINK *
|
|
|
|
|
|
|    PRC (R1 => A(TWA))
+-->+----------------------------+
    |load A(TWA)                 |
    |establish PRC RSA in TWA    |
    |load A(COMMON TABLE)from TWA|
    |a call to create a process? |
    |    .                       |
    |CRTPRC RTN:                 |
    |load A(parmlist) from TWA   |
    |search TASKLIST TABLE       |
    |if no duplicate,            |
    | build TASKLIST TABLE entry,|
    | ATTACH task                |--+
    |RETURN                      |  |
    +----------------------------+  |
                                    |
                                    |   ATTACH STUB
                                    +->+----------------------+
                                       | GETMAIN TWA, etc.    |
                                    +--| LINK to MONITOR MPU  |
                                    |  +----------------------+
                                    LINK
                                    |
                                    |   MONITOR MPU task
                                    +->+----------------------+
                                       |     .                |
                                       | CALL 'CRTMBX' USING  |
                                       |               NAME,  |
                                       |               etc.   |
                                       |                      |
                                       |     .                |
                                       | CALL 'CRTPRC' USING  |
                                       |               COMMPU,|
                                       |               etc.   |
                                       |     .                |
                                       +----------------------+
                                       | OSIIBM               |
                                       +----------------------+
```

NOTES:

The job step task (the program specified in the EXEC statement in the JCL) is the MVS Initialization program (OSIMVSI). It GETMAINS a TWA for the task, LOADs the COMMON TABLE, and then searches the MODULE LOAD TABLE, the TASK ATTACH TABLE, and the PROGRAM LINK TABLE (which are tables within the COMMON TABLE load module) to determine what modules to load and tasks to attach to prepare the region for execution.

The last (or only) program, the initialization program, links to is the NTM Monitor AP.

The NTM Monitor AP creates the Monitor MPU task and then sends it messages for it to start the MPUs required, such as the COMM MPU and the QP and PIOS MPUs.

The Monitor AP also sends messages to the COMM MPU for it to start the COMMs for the VAX and the Honeywell Level 6.

At this point the system is initialized and is ready for message traffic.

SUBTASK HIERARCHY

```
                              +----------+
                              | NTM      |
                              | MONITOR  |
                              | AP       |
                              +----------+
                                   |
                       +-------------------------+
                       |                         |
                 +----------+              +----------+
                 | ERROR    |              | NTM MPU  |
                 | LOG      |              |   for    |
                 | WRITE    |              | MONITOR  |
                 +----------+              +----------+
                                                |
          +---------------------------------------------+
          |                        |                    |
     +----------+            +-----------+         +----------+
     | NTM MPU  |            | NTM MPU   |         | NTM MPU  |
     |   for    |            |   for     |         |   for    |
     |  COMM    |            |   QPs     |         |  PIOS    |
     +----------+            +-----------+         +----------+
          |                       |                     |
   +----------+           +-----------+          +------------+
   |          |           |           |          |            |
   |          |           |  +-----------+       |       +----------+
+--------++--------+  +----------+ QP n  |  +----------+ PIOS     |
| COMM1  || COMM2  |  | QP 1     |       |  | PIOS     | PSEUDO   |
|  for   ||  for   |  |          |       |  | PSEUDO   | AP n     |
|  VAX   ||  HL6   |  |          |-------+  | AP 1     |----------+
+--------++--------+  +----------+          +----------+
```

The highest level task is the NTM Monitor AP. It receives control from the MVS initialization program (via a LINK) after the initialization program has started the Error Log Write subtask.

The NTM Monitor AP starts the Monitor MPU subtask, and the Monitor MPU starts the COMM MPU, THE QP MPU and the PIOS MPU.

The COMM MPU starts COMM1 for communication with the VAX, and COMM2 for communication with the Honeywell Level 6.

The QP MPU starts one or more QPs (Query Processors), and the PIOS MPU starts one or more PIOS Pseudo-APs.

All the tasks mentioned above run all day until told to terminate, except the QPs and PIOS Pseudo-APs. These tasks are transactions initiated by input from a terminal operator and are repeatedly started and ended all day.

APPENDIX C

VTAM LOGIC FOR IHC

The following document describes the logic and the VTAM macros used by the OSIIHC module (IHC). IHC will communicate with a 3270 terminal (or host emulating a 3270 type terminal) using a half duplex flip flop (HDFF) protocol. This was chosen over a half duplex contention protocol because its use is more prevalent with 3270 terminals. Also, there is no mechanism for communicating contention situations to the calling program (COM) since they are not apparent at the time the SEND is issued. In HDFF protocol one logical unit (LU) is in send mode, and the other is in receive mode. Their respective states are switched when the sender sends a message with a change direction indicator.

1. Initialize Logic

   A. Verify that the correct number of parameters were passed and that the event block has been initialized to zeros.

   B. Save the address of all control blocks in the TWA and move the terminal id into the TWA.

   C. If the TWA contains a nonzero pointer to the VTAM control block module (this indicates the port has already been initialized)

      1. Issue a CLSDST for the session.

      2. Close the ACB.

      3. Reset the VTAM control flags to their initial state.

      4. Delete the IHCEXT module.

      5. Go to step D.

   else

      1. Load the VTAM control block module that is associated with the port we are initializing and save its address in the TWA.

   D. Load the exit routine module (IHCEXT) which will handle error responses and abnormal conditions such as TPEND and LOSTERM. Fill in the ACB exit list with the addresses of the exit routines that are contained in IHCEXT.

   E. Obtain storage to be used as send and receive buffers.

   F. Establish a session with the remote station or in VTAM terminology, secondary logical unit (SLU).

1. **Open the ACB.**
2. **Issue a SETLOGON.**
3. **Issue an OPNDST with OPTCD=ACQUIRE.**
4. **Move the CID (session identifier) to all RPLs.**
5. **Set a flag indicating the session is in a between bracket state.**

2. Send Logic

A. Verify parameters and save the addresses of the EVTBLK and XMTBLK.

B. Verify that a receive is not outstanding.

C. Move the data from the XMTBLK to the send buffer, inserting the 3270 control characters to erase the buffer and leave the keyboard locked.

D. If between bracket flag is on

1. Issue a bid, set a timer, and wait for a definite response. If the timer expires before a response is received, we exit with a 'wait timeout error'. (Whenever IHC must wait for a reponse from the SLU, a timer is set and a wait timeout error may occur. However, no timer is set when IHC waits for VTAM to schedule an event.) If the SLU is in transmit mode, it will give a negative response and we will exit with a message indicating a bid failure.

2. Issue a SEND containing the data to be transmitted. This SEND will have a begin bracket, no end bracket, and no change direction indicators set.

3. Turn off the flag indicating the session is between brackets.

else

1. If change direction flag (VTAMCDIR) is on and the remote accepts data flow control requests

   a. Issue a session control (SESSIONC) macro with a CLEAR option.

   b. Issue a SESSIONC macro with a START DATA TRAFFIC option.

   c. Wait for a response.

        2. Issue SEND a with no begin bracket, no end bracket, no change direction and exception responses only. This send will contain the data to be transmitted.

        3. Wait for the SEND to be scheduled.

    E. Issue a SEND to erase 3270 buffer, unlock the keyboard and set the change direction indicator.

    F. Turn on the change direction flag.

    G. Set successful return code and exit.

3. Receive Logic

    A. Validate parameters and prime the EVTBLK and other areas.

    B. Ensure that the 'receive outstanding' flag is not set and that the event number is not zero.

    C. If the 'return receive' flag is set:

        1. Discard the data.

        2. Issue a SEND with erase write, change direction, and keyboard unlock.

    D. Issue an asynchronous receive, specifying that the ECB in the EVTRBLK is to be posted upon completion.

    E. Set 'receive outstanding' and event type='receive' in the EVTBLK. Move the event number from the parameter list to the EVTBLK.

    F. Set successful return code and exit.

    G. If definite response is requested by the SLU, send the appropriate positive response.

    H. Set successful return code and exit.

4. Get Logic

    A. Validate parameters.

    B. Check to see that the ECB has been posted and that the event type is 'receive'.

    C. Move the data from the buffer to the RCVBLK stripping off the 3270 control characters.

    D. Set the length field in the RCVBLK and reset the flag indicating event outstanding.

E. If the change direction flag is on, reset the change direction indicator. (We are now the sender).

F. Reset the ECB and RPL used for the receive.

G. If definite response is requested by the SLU, send the appropriate positive response.

H. Set successful return code and exit.

5. Cancel Logic

   A. Validate parameters.

   B. Insure that either a 'receive' or 'returned receive' is outstanding.

   C. Issue a RESETSR, canceling all synchronous data flows.

   D. Wait for the RESETSR to be completed.

   E. Reset the receive RPL so that it may be reused.

   F. If this terminal accepts data flow control requests:

      1. Issue a SESSIONC with a CLEAR option.

      2. Issue a SESSIONC with a START DATA TRAFFIC option.

      3. Set the flag indicating we are between brackets.

   G. Turn off the change direction flag.

   H. Set successful return code and exit.

6. Terminate Logic

   A. Verify parameters.

   B. Zero control blocks.

   C. Issue a CLSDST to end the session.

   D. Close the ACB.

   E. Delete the VTAM control block module and exit routine module.

   F. Set successful return code and exit.

7. Response Exit

   A. If response is negative

1. Format an error message and call ERRPRO to record the error.

2. Issue a SESSIONC with a CLEAR option to reinitialize the session.

3. If the SLU accepts data flow control requests, issue a SESSIONC with a START DATA TRAFFIC option. Wait for the response.

4. Turn on the between bracket flag.

5. Return to VTAM.

B. If the response is positive (this should not happen)

1. Format an error message and call ERRPRO to record it.

2. Return to VTAM.

8. TPEND Exit Routine

(VTAM has been terminated or our APPLID has been inactivated.)

A. Format an error message and call ERRPRO to record it.

B. Set the flag indicating TPEND. After this flag has been set, all calls to IHC except TRMLAN and INILAN will be rejected with an error indicating 'tpend'.

9. Losterm exit routine (line failure, buffer overflow, etc.)

A. Format an error message and call ERRPRO to record it.

B. Set the flag indicating LOSTTERM. After this flag has been set, all calls to IHC except TRMLAN and INILAN will be rejected.

C. Return to VTAM.

10. Defining parameters associated with the SLU (remote port)

Each SLU (host or terminal) will hve a load module associated with it that will contain all the VTAM control blocks necessary to establish and maintain communications. This module will have the same name as the port id. It is created by the 'VTAMBLKS' macro. For each remote port, one of these macros must be coded, assembled and link edited as re-entrant. The format of the macro is shown below.

```
VTAMBLKS APPLID=XXXX,TRMID=YYYY,VTRESID=ZZZZ,
WAIT=NNNNN
```

where

APPLID is the name of the VTAM applid. This applid must be authorized to acquire terminals. A separate applid must be defined in VTAM fcr each remote port.

TRMID is the port name that is passed by COM to the IHC program.

VTRESID is the VTAM resource id associated with the port (i.e., the label on the LU macro that defines this device.)

WAITV is the maximum time (in macro seconds) that the IHC program will wait for a response from the SLU. The default value is 10,000 or 10 seconds. It should be noted that this timeout value only applies to waiting for replies from the SLU. When it is necessary to wait for VTAM to schedule an event, there is no timeout.

Additionally, TFFLAG= 'hex value' may be specified. This option is used to specify terminal characteristics. Currently, the only terminal characteristics defined are whether or not the 'terminal' supports data flow control RU's and other characteristics of a terminal with a FM profile of 3 and a TS profile of 3. If this bit is off, the SLU is treated as a local, non-SNA 3270. The default value (on) is correct for remote 3274 and 3276 SNA terminals. If support for additional terminals is desired, this field may be expanded.